

# DW\_addsub\_dx

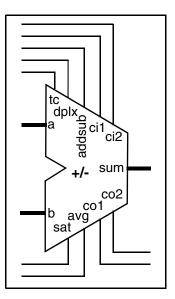
# Duplex Adder/Subtractor with Saturation and Rounding

Version, STAR and Download Information: IP Directory

### **Features and Benefits**

# **Revision History**

- Selectable single full-width Add/Sub (simplex) or two smaller width Add/Sub operations (duplex)
- Selectable saturation mode
- Selectable average mode
- Selectable number system (unsigned or twos complement)
- Parameterized full word width
- Parameterized partial word width (allowing for asymmetric partial width operations)
- Carry-out signals (one for lower half and one for full and upper half) that numerically extend the calculated sum (maintaining full precision)
- Carry-in signals (one for full and lower half and one for upper half)



# **Description**

DW\_addsub\_dx performs addition and subtraction of operands a and b as either:

- A single sum of *width* bits, or
- Two sums (one of *p1\_width* bits and one of [*width p1\_width*] bits).

The sum or difference passes through a saturation unit and then through an arithmetic shifter. The saturation unit and shifter are controlled through the input ports, sat and avg, respectively.

The two's complement select input signal, tc, indicates the processing of unsigned or signed values. When tc is low, unsigned values are processed; when tc is high, signed values are processed.

Table 1-1 Pin Description

| Pin Name | Width      | Direction | Function                  |
|----------|------------|-----------|---------------------------|
| а        | width bits | Input     | Input data                |
| b        | width bits | Input     | Input data                |
| ci1      | 1 bit      | Input     | Full or part1 carry input |
| ci2      | 1 bit      | Input     | Part2 carry input         |

## Table 1-1 Pin Description (Continued)

| Pin Name | Width      | Direction | Function                              |
|----------|------------|-----------|---------------------------------------|
| addsub   | 1 bit      | Input     | Add/subtract select input             |
|          |            |           | ■ 0 = Performs add                    |
|          |            |           | ■ 1 = Performs subtract               |
| tc       | 1 bit      | Input     | Two's complement select (active high) |
| sat      | 1 bit      | Input     | Saturation mode select (active high)  |
| avg      | 1 bit      | Input     | Average mode select (active high)     |
| dplx     | 1 bit      | Input     | Duplex mode select (active high)      |
| sum      | width bits | Output    | Output data                           |
| co1      | 1 bit      | Output    | Part1 carry output                    |
| co2      | 1 bit      | Output    | Full width or part2 carry output      |

## **Table 1-2** Parameter Description

| Parameter | Values         | Description                           |
|-----------|----------------|---------------------------------------|
| width     | ≥ 4            | Word width of a, b, and sum           |
| p1_width  | 2 to width – 2 | Word width of part1 of duplex Add/Sub |

## Table 1-3 Synthesis Implementations <sup>a</sup>

| Implementation Name | Function                            | License Feature Required |
|---------------------|-------------------------------------|--------------------------|
| rpl                 | Ripple Carry Synthesis Model        | DesignWare               |
| rpcs                | Ripple Carry Select Synthesis Model | DesignWare               |
| csm                 | Conditional Sum Synthesis Model     | DesignWare               |

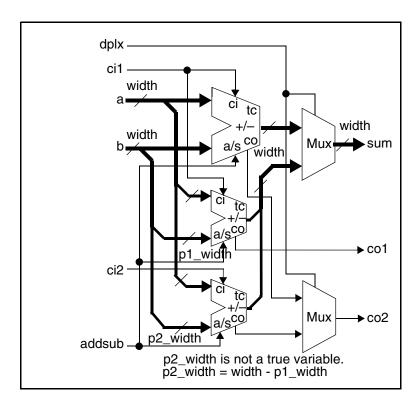
a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table.

### Table 1-4 Simulation Models

| Model                            | Function                             |
|----------------------------------|--------------------------------------|
| DW01.DW_ADDSUB_DX_CFG_SIM        | Design unit name for VHDL simulation |
| dw/dw01/src/DW_addsub_dx_sim.vhd | VHDL simulation model source code    |
| dw/sim_ver/DW_addsub_dx.v        | Verilog simulation model source code |

Figure 1-1 shows a block diagram of DW\_addsub\_dx.

Figure 1-1 DW\_addsub\_dx Block Diagram



In addition to influencing saturation and shifting operations, to influences the behavior of the carry output signals co1 and co2 such that, even for signed values, the carry outputs become a numeric extension of the sum or difference.

An example is given below:

December 2022

```
Unsigned: "1111" + "0000" = "01111"
                       0
             15
                                 15
           "1000" + "1000" = "10000"
              8
                       8
                                 16
Signed:
           "1111" + "0000" = "11111"
                       0
                                 -1
           "1000" + "1000" = "10000"
             -8
                       -8
                                -16
```

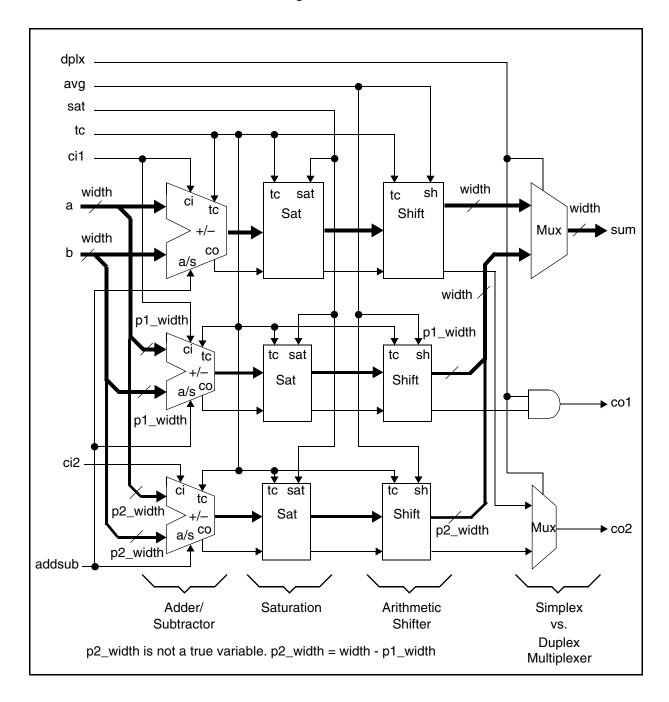
Note that in the above example, the carry output (bold digit of binary result) does not always have the same value for signed versus unsigned operation.

The addsub input determines whether addition or subtraction is performed. When addsub is low, DW\_addsub\_dx performs an add; when addsub is high, subtraction is performed.

The dplx input selects whether a single full-width operation (dplx low) or two smaller width operations (dplx high) are performed.

Figure 1-2 shows a functional block diagram of DW\_addsub\_dx.

Figure 1-2 DW\_addsub\_dx Functional Block Diagram



# Table 1-5 Operating Modes

| avg | sat | dplx | addsub | tc | Function  |
|-----|-----|------|--------|----|---|
| 0   | 0   | 0    | 0      | 0  | Simplex unsigned add operation: [co2,sum] = a + b + ci1   |
| 0   | 0   | 0    | 0      | 1  | Simplex signed add operation: [co2,sum] = a + b + ci1   |
| 0   | 0   | 0    | 1      | 0  | Simplex Unsigned subtract operation:<br>[co2,sum] = a - b - ci1   |
| 0   | 0   | 0    | 1      | 1  | Simplex signed subtract operation: [co2,sum] = a - b - ci1  |
| 0   | 0   | 1    | 0      | 0  | Duplex unsigned add operation:  [co1,p1_sum] = p1_a + p1_b + ci1 [co2,p2_sum] = p2_a + p2_b + ci2   |
| 0   | 0   | 1    | 0      | 1  | Duplex signed add operation:<br>[co1,p1_sum] = p1_a + p1_b + ci1<br>[co2,p2_sum] = p2_a + p2_b + ci2  |
| 0   | 0   | 1    | 1      | 0  | Duplex unsigned subtract operation:  [co1,p1_sum] = p1_a - p1_b - ci1 [co2,p2_sum] = p2_a - p2_b - ci2  |
| 0   | 0   | 1    | 1      | 1  | Duplex signed subtract operation:  [co1,p1_sum] = p1_a - p1_b - ci1 [co2,p2_sum] = p2_a - p2_b - ci2  |
| 0   | 1   | 0    | 0      | 0  | Simplex saturated unsigned addition operation: [co2,sum] = unsigned_saturate(a + b + ci1)   |
| 0   | 1   | 0    | 0      | 1  | Simplex saturated signed addition operation: [co2,sum] = signed_saturate(a + b + ci1)   |
| 0   | 1   | 0    | 1      | 0  | Simplex saturated unsigned subtract operation: [co2,sum] = unsigned_saturate(a - b - ci1)   |
| 0   | 1   | 0    | 1      | 1  | Simplex saturated signed subtract operation: [co2,sum] = signed_saturate(a - b - ci1)   |
| 0   | 1   | 1    | 0      | 0  | Duplex saturated unsigned addition operation:  [co1,p1_sum] = unsigned_saturate(p1_a + p1_b + ci1)  [co2,p2_sum] = unsigned_saturate(p2_a + p2_b + ci2) |
| 0   | 1   | 1    | 0      | 1  | Duplex saturated signed addition operation:  [co1,p1_sum] = signed_saturate(p1_a + p1_b + ci1)  [co2,p2_sum] = signed_saturate(p2_a + p2_b + ci2)       |
| 0   | 1   | 1    | 1      | 0  | Duplex saturated unsigned subtract operation:  [co1,p1_sum] = unsigned_saturate(p1_a - p1_b - ci1)  [co2,p2_sum] = unsigned_saturate(p2_a - p2_b - ci2) |

Table 1-5 Operating Modes (Continued)

| avg | sat | dplx | addsub | tc | Function  |
|-----|-----|------|--------|----|---|
| 0   | 1   | 1    | 1      | 1  | Duplex saturated signed subtract operation:  [co1,p1_sum] = signed_saturate(p1_a - p1_b - ci1)  [co2,p2_sum] = signed_saturate(p2_a - p2_b - ci2)                             |
| 1   | 0   | 0    | 0      | 0  | Simplex averaged unsigned addition operation:<br>[co2,sum] = (a + b + ci1) >> 1   |
| 1   | 0   | 0    | 0      | 1  | Simplex averaged signed addition operation:<br>[co2,sum] = (a + b + ci1) >> 1   |
| 1   | 0   | 0    | 1      | 0  | Simplex Averaged Unsigned Subtract operation:<br>[co2,sum] = (a - b - ci1) >> 1   |
| 1   | 0   | 0    | 1      | 1  | Simplex averaged signed subtract operation:<br>[co2,sum] = (a - b - ci1) >> 1   |
| 1   | 0   | 1    | 0      | 0  | Duplex averaged unsigned addition operation:  [co1,p1_sum] = (p1_a + p1_b + ci1) >> 1  [co2,p2_sum] = (p2_a + p2_b + ci2) >> 1  |
| 1   | 0   | 1    | 0      | 1  | Duplex averaged signed addition operation:  [co1,p1_sum] = (p1_a + p1_b + ci1) >> 1  [co2,p2_sum] = (p2_a + p2_b + ci2) >> 1  |
| 1   | 0   | 1    | 1      | 0  | Duplex averaged unsigned subtract operation:  [co1,p1_sum] = (p1_a - p1_b - ci1) >> 1  [co2,p2_sum] = (p2_a - p2_b - ci2) >> 1  |
| 1   | 0   | 1    | 1      | 1  | Duplex averaged signed subtract operation:  [co1,p1_sum] = (p1_a - p1_b - ci1) >> 1  [co2,p2_sum] = (p2_a - p2_b - ci2) >> 1  |
| 1   | 1   | 0    | 0      | 0  | Simplex averaged saturated unsigned addition operation: [co2,sum] = (unsigned_saturate(a + b + ci1)) >> 1   |
| 1   | 1   | 0    | 0      | 1  | Simplex averaged saturated signed addition operation:  [co2,sum] = (signed_saturate(a + b + ci1)) >> 1  |
| 1   | 1   | 0    | 1      | 0  | Simplex averaged saturated unsigned subtract operation: [co2,sum] = (unsigned_saturate(a - b - ci1)) >> 1   |
| 1   | 1   | 0    | 1      | 1  | Simplex averaged saturated signed subtract operation: [co2,sum] = (signed_saturate(a - b - ci1)) >> 1   |
| 1   | 1   | 1    | 0      | 0  | Duplex averaged saturated unsigned addition operation:  [co1,p1_sum] = (unsigned_saturate(p1_a + p1_b + ci1)) >> 1 [co2,p2_sum] = (unsigned_saturate(p2_a + p2_b + ci2)) >> 1 |

**Table 1-5** Operating Modes (Continued)

| avg | sat | dplx | addsub | tc | Function   |
|-----|-----|------|--------|----|--|
| 1   | 1   | 1    | 0      | 1  | Duplex averaged saturated signed addition operation:  [co1,p1_sum] = (signed_saturate(p1_a + p1_b + ci1)) >> 1  [co2,p2_sum] = (signed_saturate(p2_a + p2_b + ci2)) >> 1       |
| 1   | 1   | 1    | 1      | 0  | Duplex averaged saturated unsigned subtract operation:  [co1,p1_sum] = (unsigned_saturate(p1_a - p1_b - ci1)) >> 1  [co2,p2_sum] = (unsigned_saturate(p2_a - p2_b - ci2)) >> 1 |
| 1   | 1   | 1    | 1      | 1  | Duplex averaged saturated signed subtract operation:  [co1,p1_sum] = (signed_saturate(p1_a - p1_b - ci1)) >> 1  [co2,p2_sum] = (signed_saturate(p2_a - p2_b - ci2)) >> 1       |

# **Related Topics**

- Math Arithmetic Overview
- DesignWare Building Block IP User Guide

```
library IEEE, DWARE;
use IEEE.std logic 1164.all;
use DWARE.DW Foundation comp arith.all;
entity DW addsub dx inst is
  generic (inst width
                          : NATURAL := 24;
            inst p1 width : NATURAL := 16 );
  port (inst a
                    : in std logic vector(inst width-1 downto 0);
                     : in std logic vector(inst width-1 downto 0);
         inst b
         inst cil
                    : in std logic;
                    : in std logic;
         inst ci2
         inst addsub : in std logic;
         inst tc : in std logic;
         inst sat
                    : in std logic;
         inst avq : in std logic;
         inst_dplx : in std_logic;
         sum_inst : out std_logic_vector(inst_width-1 downto 0);
         col inst
                     : out std logic;
         col_inst : out std_logic;
co2_inst : out std_logic );
end DW addsub dx inst;
architecture inst of DW addsub dx inst is
begin
  -- Instance of DW addsub dx
  U1 : DW addsub dx
    generic map ( width => inst width, pl width => inst pl width )
    port map ( a => inst a, b => inst b,
               ci1 => inst ci1, ci2 => inst ci2, addsub => inst addsub,
               tc => inst tc, sat => inst sat, avg => inst avg,
               dplx => inst dplx, sum => sum inst,
               co1 => co1 inst, co2 => co2 inst );
end inst;
-- pragma translate off
configuration DW addsub dx inst cfg inst of DW addsub dx inst is
  for inst
  end for; -- inst
end DW addsub_dx_inst_cfg_inst;
-- pragma translate on
```

# **HDL Usage Through Component Instantiation - Verilog**

```
module DW addsub dx inst( inst a, inst b, inst ci1, inst ci2, inst addsub,
                          inst tc, inst sat, inst avg, inst dplx, sum inst,
                          col inst, co2 inst );
 parameter width = 24;
 parameter p1 width = 8;
  input [width-1 : 0] inst a;
  input [width-1:0] inst b;
  input inst ci1;
  input inst_ci2;
  input inst addsub;
  input inst tc;
  input inst sat;
  input inst avq;
  input inst dplx;
  output [width-1:0] sum inst;
  output co1 inst;
  output co2 inst;
 // Instance of DW addsub dx
 DW addsub dx #(width, p1 width)
   U1 ( .a(inst a), .b(inst b), .ci1(inst ci1), .ci2(inst ci2),
         .addsub(inst addsub), .tc(inst tc), .sat(inst sat),
         .avg(inst_avg), .dplx(inst_dplx),
         .sum(sum inst), .co1(co1 inst), .co2(co2 inst));
```

endmodule

# **Revision History**

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

| Date       | Release       | Updates   |
|------------|---------------|---|
| March 2019 | DWBB_201903.0 | ■ Added this Revision History table and the document links on this page |

## **Copyright Notice and Proprietary Information**

© 2022 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

### **Destination Control Statement**

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

#### **Disclaimer**

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

#### **Trademarks**

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

## Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

## **Third-Party Links**

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc. www.synopsys.com